Estudo de Técnicas de Aprendizado Baseado em Uma Única Classe para o Reconhecimento Facial

Leonardo G. de Moraes[1], **Rafael G. Rossi**[2]

[1] Federal University of Mato Grosso do Sul (UFMS)
Campus Três Lagoas
Av. Ranulpho Marques Leal, 3484, CEP 79613-000, Fone: 67 35093750

leonardogmoraes2000@gmail.com, rafael.g.rossi@ufms.br

Resumo. *O reconhecimento facial (RF) é uma técnica de autenticação biométrica que não exige contato, permitindo que um sistema computacional reconheça um indivíduo (classe) tanto próximo quanto distante. Normalmente, o RF se dá através da extração e de características dos rostos dos indivíduos, e sempre que uma autenticação é necessária, é feita uma busca por similaridades (matching) com os dados dos rostos previamente salvos. Tal abordagem pode ser muito lenta, principalmente em grandes bases de dados. Para agilizar o processo de classificação, podem ser utilizadas técnicas de aprendizado de máquina (AM) para extrair características de imagens, além de gerar modelos para automatizar a classificação de indivíduos. Entretanto, a maioria dos trabalhos considera algoritmos de aprendizado de máquina multi-classe (AMMC), categoria essa em que é necessário fornecer ao modelo exemplos de todas as classes a serem classificadas. Em sistemas de RF em que novos usuários tendem a ser inseridos constantemente, os modelos de aprendizado de máquina devem ser retreinados constantemente, o que também pode demandar tempo e não possibilitar a autenticação de um novo usuário de maneira imediata. Para lidar com tal problema, pode-se fazer o uso do aprendizado de máquina baseado em uma única classe (AMUC), tipo de AM em que apenas a classe de interesse é informada ao algoritmo, podendo o mesmo, após o aprendizado, discriminar a classe de interesse das demais classes. Dado isso, o objetivo deste trabalho é analisar o uso de algoritmos de AMUC para o RF. Para isso, foram utilizados os algoritmos de autoencoders convolucionais, os quais são adequados para o AMUC para imagens, além da proposta do One-Class Local Binary Patterns Histogram (OC-LBPH), uma versão AMUC do algoritmo Local Binary Patterns Histogram (LBPH). Foram realizados experimentos em 10 bases de dados, contendo, no total, 12588 imagens de faces de 797 pessoas diferentes. Foi utilizado o procedimento 10-fold cross validation e calculada a acurácia dos resultados para avaliar os algoritmos, e os resultados dos métodos AMUC foram comparados aos dos algoritmos AMMC e de matching. A partir dos experimentos realizados, a proposta do OC-LBPH obteve os melhores resultados em 6 das 10 coleções de dados.*

1. Introdução

Ao longo dos últimos anos, o número de usuários de sistemas computacionais vem crescendo cada vez mais [Max Roser and Ortiz-Ospina 2015], o que resulta em uma

crescente quantidade de dados sendo gerados. Para evitar que os dados desses usuários sejam indevidamente acessados, se faz necessária uma autenticação, i.e., um controle de acesso [Wechsler et al. 2012]. Esse controle vale, não apenas para o acesso a sistemas computacionais ou dispositivos como celulares, mas também para recintos, como aeroportos, universidades, empresas, entre outros.

Sistemas computacionais podem autenticar os usuários por meio de três abordagens diferentes: (i) a partir de algo que o usuário sabe, i.e., *login* e senha ou respostas a perguntas predefinidas; (ii) a partir de algo que o usuário tem, como chaves, *tokens*, cartões de acesso, e outros; (iii) a partir de algo que usuário é [Weaver 2006], i.e., biometria, como impressões digitais, reconhecimento por íris ou face. O reconhecimento facial (RF), apesar de possuir uma menor acurácia entre as outras formas de biometria, possui a vantagem de não ser um método invasivo e não exigir contato algum, visto que é feito a partir de imagens digitais capturadas por câmeras que não precisam estar próximas ao indivíduo [Li and Jain 2011], podendo um sistema computacional reconhecer vários indivíduos de uma única vez com um único equipamento.

Uma das formas comuns de se realizar o RF se dá através da extração de características das faces dos indivíduos, que então são armazenadas em um banco de dados (BD) [Guo and Zhang 2019]. Então, sempre que uma autenticação é necessária, é realizada uma busca por similaridades (*matching*) com os dados armazenados no BD com o intuito de encontrar as características das faces armazenadas que mais se assemelham às do novo rosto apresentado ao sistema computacional. Entretanto, buscas por similaridades podem ser lentas, especialmente em sistemas computacionais em que a base de dados é muito grande, fazendo com que o reconhecimento seja mais lento e menos interessante para fins práticos.

Um campo científico que pode agilizar o processo de RF é o de visão computacional (VC), um campo de estudo que visa desenvolver técnicas para que computadores possam "ver"e compreender o conteúdo de imagens ou vídeos digitais [Forsyth and Ponce 2002]. A habilidade de identificar e classificar os objetos nas imagens é possibilitada pelo uso de modelos de classificação gerados por meio de algoritmos de aprendizado de máquina (AM). Uma das técnicas de AM que têm ganhado destaque nos últimos anos na área de VC é a de aprendizado profundo, ou, redes neurais profundas (RNP) [Aggarwal 2018].

Entretanto, quando aplicados algoritmos de AM para classificação de imagens e RF, a maioria dos trabalhos utiliza algoritmos de aprendizado de máquina multiclasse (AMMC) [Abate et al. 2007, Zhao et al. 2003]. Uma característica dessa abordagem é que, para essa categoria de algoritmos, é necessário fornecer exemplos rotulados de todas as classes, ou seja, faces a serem classificadas, para que então, após o treinamento de um modelo de classificação, novos exemplos não rotulados das classes previamente apresentadas durante o treinamento do modelo possam ser classificados [Tan et al. 2019]. Quando se trata de sistemas de RF, como câmeras de vigilância ou controle de acesso a locais, novos usuários tendem a ser inseridos a todo momento. Neste caso, um modelo de classificação de AMMC teria que ser continuamente treinado, o que pode ser muito custoso computacionalmente. Outro exemplo de aplicação de um sistema de RF é para o desbloqueio de aparelhos celulares. Nesta situação, o usuário fornece ao sistema apenas exemplos de sua face, não havendo portanto

contra-exemplos para que o AMMC possa ser aplicado.

Para lidar com situações como as dos cenários acima citados, onde para aplicações práticas da classificação ou autenticação por meio do RF é necessário apenas o fornecimento de exemplos da classe de interesse, pode-se fazer o uso do aprendizado de máquina baseado em uma única classe (AMUC) [Alam et al. 2020, Ruff et al. 2018, Kemmler et al. 2013, Li et al. 2009, Chandola et al. 2009, Tax 2002]. Nessa categoria, apenas exemplos da classe de interesse, ou, classe alvo, são apresentados para o treinamento do modelo de AM, que será então, após este treinamento, capaz de discriminar a classe alvo das demais classes que forem apresentadas. Além disso, sempre que necessária a inserção de um novo usuário, apenas será necessário ao modelo realizar o treinamento com exemplos da nova classe, tornando mais rápido o aprendizado.

Com isso, o objetivo geral deste trabalho de conclusão de curso é realizar comparações e avaliações entre os diferentes métodos apresentados de RF, comparando os algoritmos de AMUC com os de AMMC e *matching*, para assim poder apresentar uma análise de custo-benefício entre eles. A partir dos experimentos realizados, a proposta do algoritmo *One-Class Local Binary Patterns Histogram* (OC-LBPH) apresentou os melhores resultados em 6 das 10 coleções de dados. Além disso, este trabalho também visa levantar o melhor método para a implementação de um mecanismo de controle de acesso ao Laboratório de Inovação e Engenharia de Software da UFMS/CPTL em trabalhos futuros.

O restante deste trabalho de conclusão de curso está dividido da seguinte forma. Na Seção 2 são apresentados os conceitos necessários para a compreensão deste trabalho. Na Seção 3 são apresentadas as especificações da metodologia utilizada na pesquisa. Na Seção 4 são apresentados os resultados obtidos. Por fim, na Seção 5, são apresentadas as conclusões e considerações acerca de trabalhos futuros.

2. Conceitos

Nesta Seção são apresentados conceitos importantes para o desenvolvimento e compreensão deste trabalho. É explicado como são representadas imagens no meio digital, os conceitos e ideias por trás das técnicas de *matching*, redes neurais profundas e são explicados os algoritmos utilizados em cada uma dessas categorias.

2.1. Representação de imagens

No meio digital imagens são compostas por elementos chamados de pixel, que adotam cada um valor único que representa sua intensidade em um dado espaço de cor. Espaços de cor são um modelo adotado para facilitar a especificação de cores nesse meio em que cada cor é representada por um único ponto (pixel). Em sua essência, um espaço de cor é uma especificação de um sistema de coordenadas e um subespaço dentro desse sistema [Gonzalez and Woods 2018].

No caso das imagens em escala de cinza, cada pixel pode assumir valores reais no intervalo $[0,1]$ ou 256 valores inteiros no intervalo $[0..255]$, representando a intensidade da informação, compondo uma imagem em tons de cinza. A escala é apresentada na Figura 1.

Já imagens coloridas são usualmente representadas em um modelo chamado de RGB, um acrônimo para *Red, Green, Blue* (vermelho, verde, azul), um modelo ba-

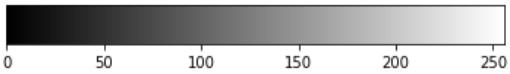
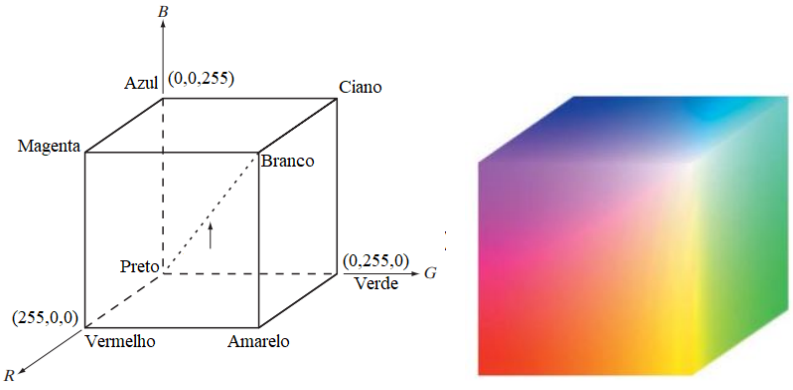


Figura 1. Exemplo das cores assumidas por um pixel com um espaço de cores de uma dimensão. Fonte: autoria própria.

seado no sistema cartesiano apresentado na Figura 2(a). Nesse modelo, 3 valores referentes às dimensões R, G e B, assumem valores no intervalo [0..255] ou [0,1]. Além do RGB, também existem outros espaços de cores, como o *cyan, magenta, yellow and key* (CMYK) ou *hue, saturation, intensity* HSI [Gonzalez and Woods 2018].



(a) Disposição de valores no espaço RGB. (b) Cubo RGB Colorido.

Figura 2. Exemplo de valores de pixels em cada cor (a) e a representação em 3 dimensões do espaço de cores RGB (b). Fonte: Adaptado de [Gonzalez and Woods 2018].

Sendo assim, podemos assumir uma imagem digital como sendo um tensor de três dimensões onde a terceira dimensão representa as cores de cada pixel, tendo a terceira dimensão apenas uma posição em imagens em escala de cinza e três posições em imagens RGB. A composição de uma imagem é exemplificada na Figura 3.

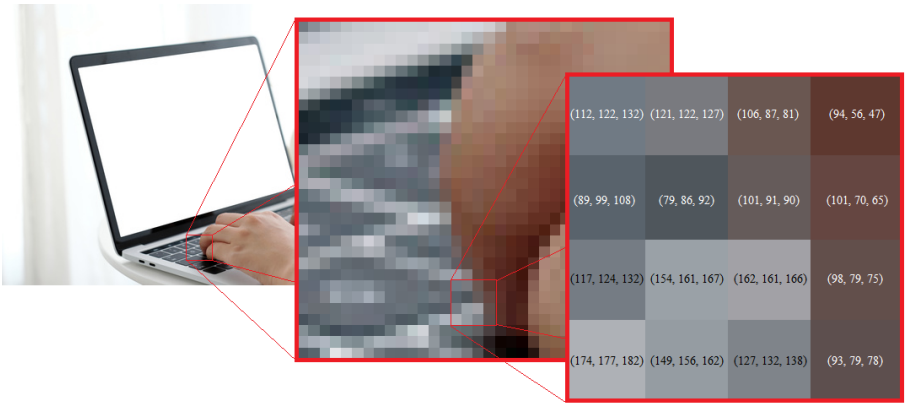


Figura 3. Exemplo de composição de uma imagem, mostrando seus pixels e os valores que possuem. Fonte: autoria própria.

2.2. Algoritmos de matching

O processo de *matching* consiste em alimentar um sistema com as imagens de rostos as quais se tem a intenção de identificar posteriormente. Para isso, é realizado um pré-processamento e as características dessas imagens são extraídas e salvas em um BD. Quando um novo rosto é informado para realização do RF, o extrator de características extrai atributos da nova imagem apresentada da mesma forma que ocorreu com as imagens previamente apresentadas ao sistema. Então, são comparadas as características extraídas com as características armazenadas no BD, para assim, corresponder a face com um indivíduo. [Guo and Zhang 2019].

Nas subseções abaixo são apresentados os três algoritmos de *matching* utilizados neste trabalho e suas abordagens para extração de características, são eles: (i) Eigenfaces; (ii) Fisherfaces; e (iii) Local Binary Patterns Histogram (LBPH). Todos pertencem à biblioteca de VC OpenCV (*Open Source Computer Vision*) [Bradski 2000]

2.2.1. Eigenfaces

O algoritmo Eigenfaces se baseia na ideia de que, na maioria das imagens de rostos, a face está contida em um subespaço vetorial muito menor do que o espaço total da imagem, ou seja, a maior parte da imagem não é importante para o RF, mas sim apenas as principais características. Portanto, é feito o uso da análise de componentes principais, em inglês, *Principal Component Analysis* (PCA) para transformar conjuntos de variáveis possivelmente correlacionadas das imagens em conjuntos menores de variáveis não correlacionadas contendo os componentes principais, também chamados de *eigenvectors*. A análise dos *eigenvectors* é feita da seguinte forma [Turk and Pentland 1991]:

Seja $X = \{x_1, x_2, ..., x_n\}$, ou seja, um vetor com n imagens de rostos na forma de vetores. É calculado o rosto médio μ.

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

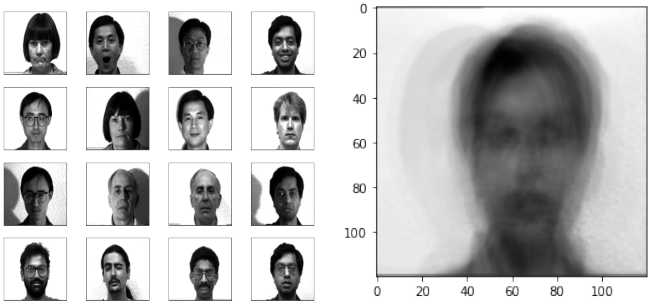
Então é calculada a matriz de covariância S.

$$S = \frac{1}{n}\sum_{i=1}^{n} (x_i - \mu)(x_i - \mu)^T$$

São calculados os *eigenvalues* λ_i e *eigenvectors* v_i da matriz S.

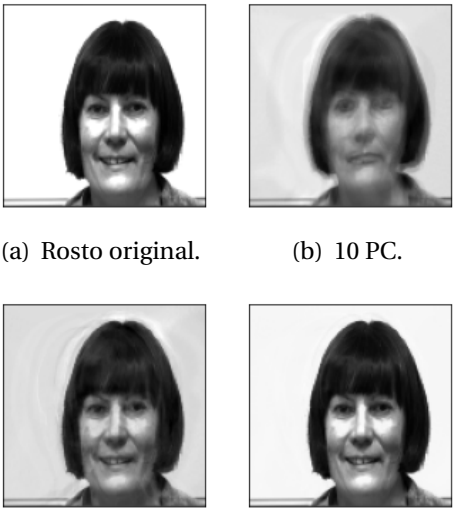
$$Sv_i = \lambda_i v_i, i = 1, 2, ..., n$$

Os *eigenvectors* são ordenados de forma decrescente por seus *eigenvalues*. Sendo o conjunto de *eigenvectors* $V = \{v_1, v_2, ..., v_k\}$, os k principais componentes (PC) de um vetor (imagem) x são dados por: $y = V^T(x - \mu)$, enquanto a reconstrução do PCA é dada por: $x = Vy + \mu$ [Duda et al. 2012, Turk and Pentland 1991].



(a) Coleção de rostos. (b) Rosto médio.

Figura 4. Exemplo de um conjunto de imagens de rostos (a) e o resultado do cálculo do rosto médio (b). Fonte: autoria própria.



(a) Rosto original. (b) 10 PC.

(c) 30 PC. (d) 50 PC.

Figura 5. Exemplo de reconstrução de um rosto (a) a partir dos 10 (b), 30 (c) e 50 (d) componentes principais. Fonte: autoria própria.

Por fim, o algoritmo Eigenfaces realiza o RF da seguinte maneira: (i) todo o conjunto de treino é projetado no subespaço PCA; (ii) a imagem a ser classificada é projetada no subespaço PCA; então, (iii) é encontrado o vizinho mais próximo entre as imagens de treino e a imagem a ser classificada.

Na Figura 4 é apresentado um exemplo do resultado de um cálculo de rosto médio a partir de uma coleção de imagens de rostos. Já na Figura 5, são apresentados os resultados da reconstrução da imagem de um dado rosto com diferentes quantidades de PC: (i) 10; (ii) 30; e (iii) 50.

Vale ressaltar que o PCA é computado com base nas imagens de treinamento corrente. Caso novas imagens ou novas faces fossem inseridas na base de dados, os *eigenvalues* e *eigenvectors* do PCA teriam que ser obtidos novamente. Além disso, esse algoritmo apresenta sensibilidade à luminosidade e ao alinhamento de elementos da face, como olhos e boca, em todas as imagens. Apesar do algoritmo Eigenfaces ser utilizado em conjunto com o algoritmo dos vizinhos mais próximos, o novo espaço

gerado pelos *eigenvectors* pode ser utilizado no treinamento de outros algoritmos de AM, como o *Support Vector Machines* (SVM).

2.2.2. Fisherfaces

O algoritmo Fisherfaces, assim como o Eigenfaces, visa realizar uma redução de dimensionalidade [Belhumeur et al. 1997]. Entretanto, diferente do algoritmo anterior, o Fisherfaces utiliza um método para redução específica por classe, i.e., maximiza a razão entre as dispersões entre as classes e as dispersões dentro das classes. Isso é realizado através do discriminante linear de Fisher, em inglês, *Fisher's Linear Discriminant* (FLD)[Fisher 1936], um método de redução de dimensionalidade supervisionado, diferentemente do PCA que é um método não supervisionado [Cheng et al. 2004]. O algoritmo funciona da seguinte forma [Belhumeur et al. 1997]:

Seja X um vetor com imagens de c classes $X = \{X_1, X_2, ..., X_c\}$ e $X_i = \{x_{c_1}, x_{c_2}, ..., x_{c_i}\}$. São calculadas a média total μ e a média de cada classe μ_i, sendo $i \in \{1, 2, ..., c\}$.

$$\mu = \frac{1}{n} \sum_{i=1}^{n} x_i$$

$$\mu_i = \frac{1}{|X_i|} \sum_{x_j \in X_i} x_j$$

Sendo n o número de imagens, as matrizes de dispersão dentro das classes S_W e entre as classes S_B são calculadas.

$$S_B = \sum_{i=1}^{c} X_i (\mu_i - \mu)(\mu_i - \mu)^T$$

$$S_W = \sum_{i=1}^{c} \sum_{x_j \in X_i} (x_j - \mu_i)(x_j - \mu_i)^T$$

Uma vez calculados S_B e S_W, é feita então a projeção ótima W_{opt} que maximiza o critério de separabilidade de classe:

$$W_{opt} = \arg\max_W \frac{|W^T S_B W|}{|W^T S_W W|}$$

Então, o PCA é utilizado para projetar a coleção de imagens em um espaço dimensional menor W_{opt}, dado por $W_{opt}^T = W_{fld}^T W_{pca}^T$, sendo

$$W_{pca} = \arg\max |W^T S_T W|$$

$$W_{fld} = \arg\max_W \frac{|W^T W_{pca}^T S_B W_{pca} W|}{|W^T W_{pca}^T S_W W_{pca} W|}$$

A classificação da imagem é realizada da mesma forma descrita no algoritmo Eigenfaces, com a classe sendo definida pelo vizinho mais próximo da nova imagem dentro do subespaço PCA.

2.2.3. Local Binary Patterns Histogram

O algoritmo Local Binary Patterns Histogram (LBPH) é baseado no algoritmo de padrões binários locais, em inglês, *local binary patterns* (LBP) é uma técnica de análise de texturas que visa resumir a estrutura de uma imagem através de comparações dos pixels com os pixels vizinhos [Ojala et al. 1996].

A partir de um raio r que delimita a vizinhança de um dado pixel central, e sendo P o número de vizinhos nesse limiar, o valor da intensidade I de um pixel denotado por x_{ij}, sendo i e j as coordenadas do pixel na imagem, é comparado com o de seus P vizinhos. Os vizinhos com intensidade maior ou igual que o pixel central recebem o valor 1 e o restante recebe o valor 0. É então realizada uma multiplicação do valor dos vizinhos (0 ou 1) com a potência de 2 da posição p que ele ocupa em relação ao pixel central. Por fim, os valores são somados e o valor do pixel central é definido. O processo é exemplificado na Figura 6. A função é descrita da seguinte forma:

$$LBP(X_{ij}) = \sum_{p=0}^{P-1} 2^p s(I_p - I_X)$$

sendo,

$$s(x) = \begin{cases} 1, & \text{se } x \geq 0 \\ 0, & \text{se } x < 0 \end{cases}$$

A imagem gerada pelo operador LBP (Figura 7(b)), é dividida em m regiões locais e então é extraído o histograma de cada região. Estes histogramas são então concatenados, gerando um só histograma de características espacialmente aprimorado, representando a imagem do rosto de forma eficiente [Ahonen et al. 2004].

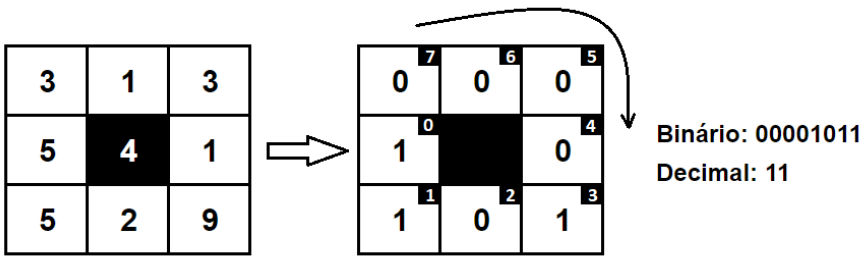
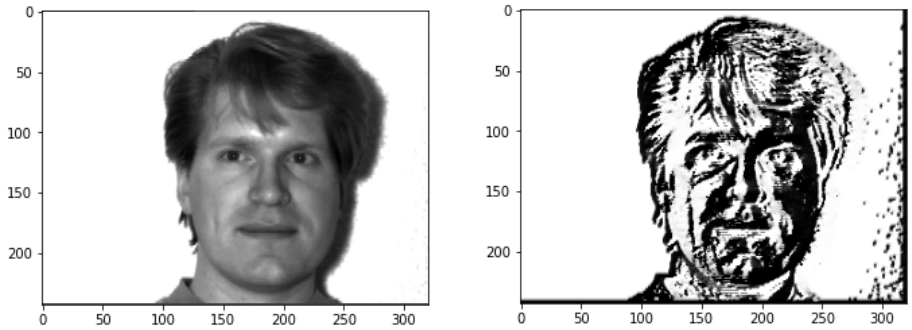


Figura 6. Exemplo do operador LBP. Fonte: autoria própria.

Enfim, as características resultantes desse histograma de padrões binários locais, em inglês, *local binary patterns histogram* são utilizadas como descritores da ima-



(a) Rosto original. (b) Imagem gerada por LBP.

Figura 7. Comparação de um rosto com a sua imagem gerada através do operador LBP. Fonte: autoria própria.

gem e, consequentemente, nas comparações para descobrir a classe da imagem que mais se assemelha à imagem apresentada para o RF.

2.3. Redes neurais profundas

As RNP são consideradas o estado-da-arte para tarefas de VC [Simonyan and Zisserman 2014]. São arquiteturas de redes neurais (RN) [Yegnanarayana 2009] que possuem múltiplas camadas escondidas, chamadas de camadas intermediárias. As camadas escondidas de uma RNP podem possuir diversas funcionalidades, como camadas de neurônios totalmente conectados (camadas densas), camadas que atuam como filtros de imagens, entre outras. Com isso, diversas modalidades de RNP podem ser arquitetadas [Aggarwal et al. 2018].

Nas próximas subseções são apresentadas as duas arquiteturas de RNP consideradas neste trabalho e suas abordagens para a classificação de imagens. São elas: (i) Redes Neurais Convolucionais (RNC), uma abordagem de AMMC; e (ii) Autoencoders, uma abordagem de AMUC.

2.3.1. Redes neurais convolucionais

Mesmo com diversas arquiteturas diferentes na literatura, as RNC utilizam três principais tipos de camadas escondidas, ou seja, camadas entre as camadas de entrada e saída de dados, são elas as camadas de convolução (de onde vem o nome da arquitetura), *pooling*, uma camada de *flattening* e, por fim, um conjunto de camadas densas [Aggarwal et al. 2018, Gu et al. 2018].

A camada de entrada, ou, *input*, de uma RN depende dos dados, após seu pré-processamento, que servirão como entrada da RN [Aggarwal et al. 2018], podendo esta camada assumir a dimensionalidade necessária para tais dados (Figura 9). No caso deste trabalho, a camada de entrada das RNs possui as dimensões das imagens de rostos após a padronização descrita na Seção 3.

As camadas de convolução atuam como filtros dos dados de entrada. São chamadas assim pois estas camadas convolvem os dados de entrada, ou seja, extraem mapas de características das imagens. Isto é feito a partir de uma operação de produto

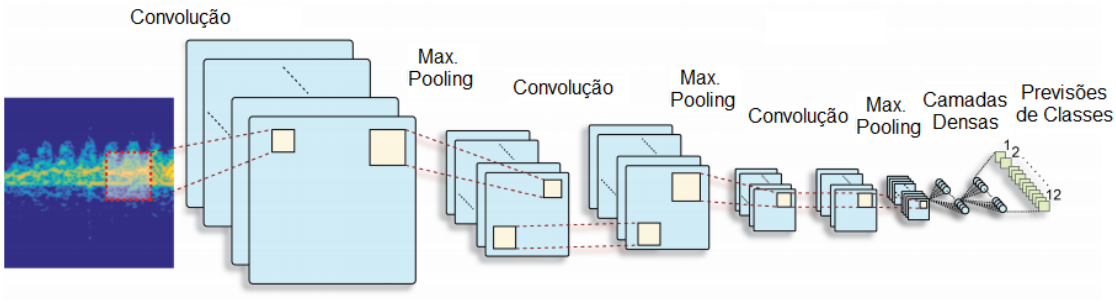


Figura 8. Exemplo de arquitetura de uma RNC. Fonte: adaptado de [Seyfioğlu et al. 2018]

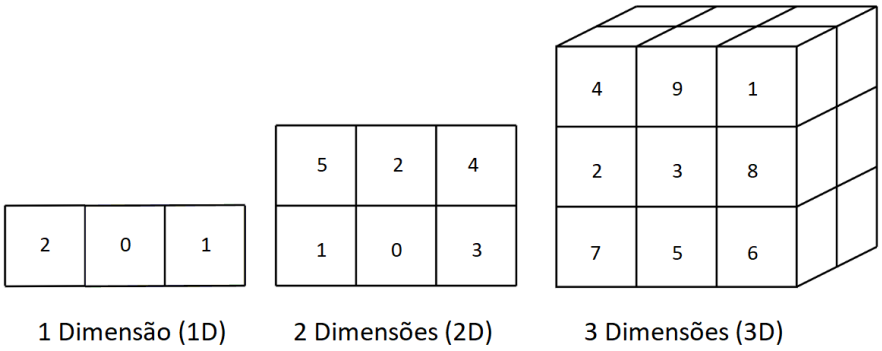


Figura 9. Diferentes dimensões de dados. Fonte: autoria própria.

escalar entre matriz de entrada da camada de convolução com uma matriz de valores chamada de *kernel* ou filtro (Figura 10), que possui o mesmo número de dimensões da matriz de entrada [Aggarwal et al. 2018]. A variação dos valores do filtro é responsável por detectar diferentes tipos de atributos em uma imagem, como exemplificado na Figura 11. No caso de uma imagem colorida, a convolução será aplicada em nas matrizes correspondentes a cada canal de cor.

As convoluções que ocorrem na rede são intercaladas com camadas de *pooling*. As operações de *pooling* selecionam características importantes e eliminam ruídos dos dados recebidos pela camada de convolução, além de reduzirem a dimensionalidade de acordo com o tamanho T da janela de *pooling*. Esta redução pode ser feita de diferentes formas (Figura 12), como o *max-pooling* [Boureau et al. 2010], que agrupa os pixels de uma matriz de tamanho T em apenas um pixel com o maior valor entre a matriz original. Outra forma comum é o *average-pooling* [Wang et al. 2012], que agrupa os pixels de uma janela de tamanho T em um só, cujo valor é o resultado da média dos valores originais [Aggarwal et al. 2018, Gu et al. 2018].

Após as camadas de convolução e de *pooling*, os dados n-dimensionais são "achatados"na camada de *flattening*. Essas camadas são responsáveis por remodelar os dados em uma única dimensão, possibilitando a ação das camadas densas que vêm logo adiante.

Após o achatamento, normalmente existe uma ou mais camadas densas, ou

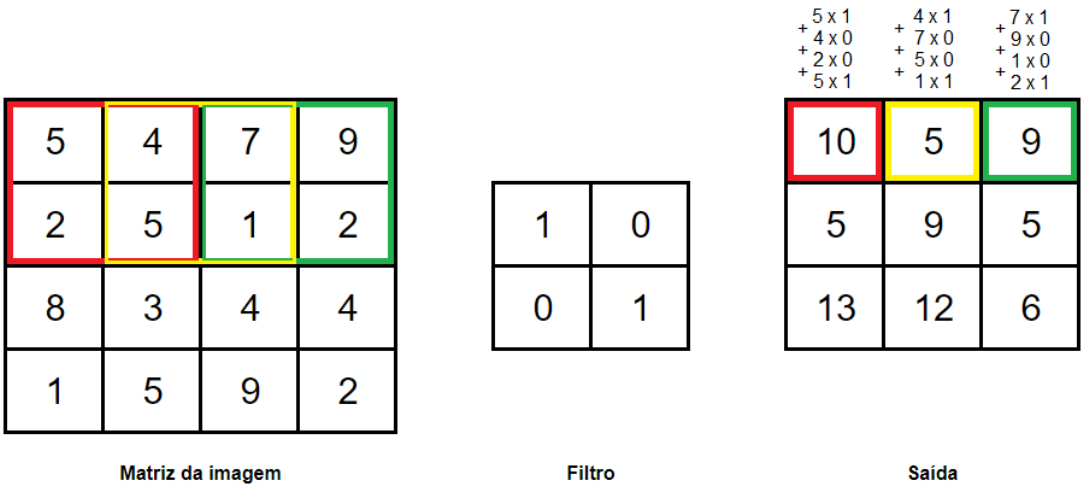


Figura 10. Exemplo de execução de convolução em uma matriz 4x4 com filtro 2x2. Fonte: autoria própria.

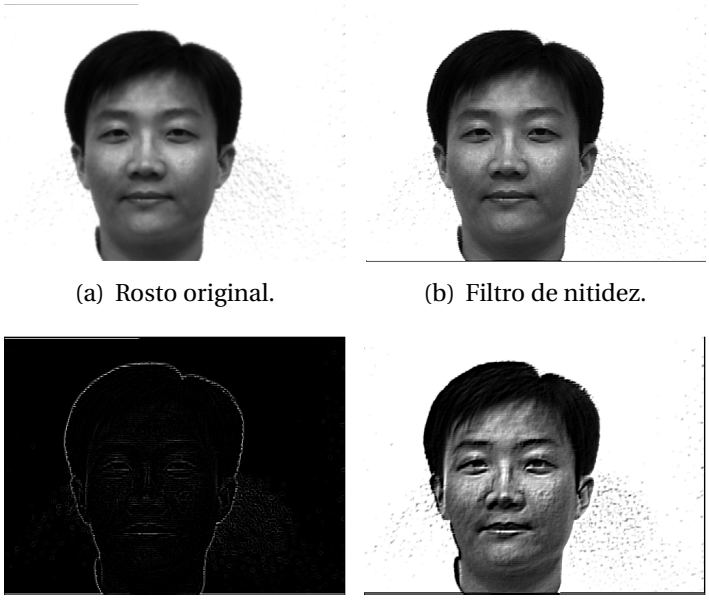
seja, camadas em que todos os neurônios estão conectados com as camadas adjacentes, como exemplificado na Figura 13. Estas camadas aprendem as superfícies de separação dos dados das diferentes classes a partir dos dados recebidos pelos filtros das camadas anteriores [Gu et al. 2018]. As camadas densas treinam com base nos parâmetros anteriormente extraídos para, enfim, realizar a classificação.

A camada de saída de uma RNC é desenhada para a aplicação a qual a RN está sendo utilizada. Para problemas de classificação, a última camada é uma camada densa com a quantidade de neurônios referentes à quantidade de classes a serem distinguidas pela RNC (Figura 13), com exceção de problemas de classificação binária, onde pode-se utilizar um único neurônio [Aggarwal et al. 2018].

2.3.2. Autoencoder

Autoencoder (AE) é uma arquitetura de RNP que visa reconstruir na camada de saída os dados que recebe na camada de entrada [Aggarwal et al. 2018]. A ideia principal é que os dados de entrada sejam mapeados em representações comprimidas nas camadas intermediárias, até que cheguem em sua menor representação dentro da rede, na camada chamada de *bottleneck* (gargalo), cuja função é aprender uma representação de menor dimensionalidade, em que o número de dimensões corresponde ao número de neurônios na camada localizada ao centro da arquitetura. Esta representação é, então, mapeada novamente em camadas maiores até chegar na saída, que possui as mesmas dimensões da entrada [Vincent et al. 2008, Larochelle et al. 2007] (Figura 14). O objetivo é tentar reconstruir todas as dimensões exatamente como eram antes da compressão.

Existem diversas arquiteturas propostas de AE, como o AE esparso [Ng et al. 2011] ou AE variacional [Kingma and Welling 2013]. Entre elas, está a arquitetura considerada neste trabalho, o AE convolucional (AEC). O AEC possui o mesmo princípio da forma tradicional do AE, reconstruir os dados de entrada na



(a) Rosto original. (b) Filtro de nitidez.

(c) Filtro de contorno. (d) Filtro de relevo.

Figura 11. Diferentes saídas resultantes de diferentes filtros de convolução. Fonte: autoria própria.

camada de saída, passando por camadas que comprimem os dados e geram uma representação reduzida. Sua principal diferença em relação ao AE tradicional é que este possui um processo de filtragem de características relevantes. Isto é feito utilizando filtros de convolução antes do aprendizado de representação, assim como as RNC. Já para a reconstrução, é realizada a operação inversa, uma convolução transposta, a fim de mapear a representação comprimida em sua forma original [Aggarwal et al. 2018], como exemplificado nas Figuras 15(a) e 15(b).

Desta forma, o AEC aprende baseando-se apenas na classe de interesse, possibilitando o AMUC. Para o RF, a ideia é que o AEC seja capaz de reconstruir qualquer imagem do rosto da classe de interesse após o treinamento sem a utilização de contra-exemplos.

3. Método de Pesquisa

Para atingir os objetivos deste trabalho, o método de pesquisa foi dividido nas seguintes partes: (i) coleta de bases de dados de rostos rotulados; (ii) implementação e execução de técnicas de *matching*; (iii) implementação e execução de AMMC, com diferentes RNCs; (iii) implementação e execução de AMUC, com os algoritmos propostos de AEC e OC-LBPH; e (iv) análise dos resultados. Nas próximas seções são apresentados os resultados de cada uma dessas etapas.

3.1. Conjuntos de Dados

Como etapa inicial e essencial para as etapas subsequentes, foram coletadas as seguintes bases de imagens de rostos rotulados: MIT CBCL Face Recognition Database [Weyrauch et al. 2004], Faces 1999 (Front) [Weber 1999], Faces 94, Faces 95, Faces 96, Grimace [Spacek 2008], Georgia Tech face database [Nefian 2013], Yale Face Database

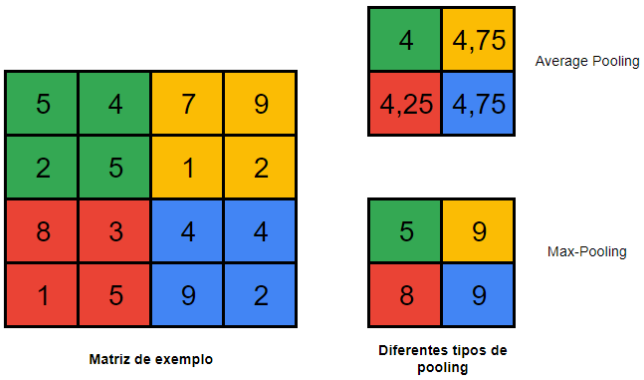


Figura 12. Exemplo de diferentes tipos de *pooling* com uma janela de $T = 2$. Fonte: autoria própria.

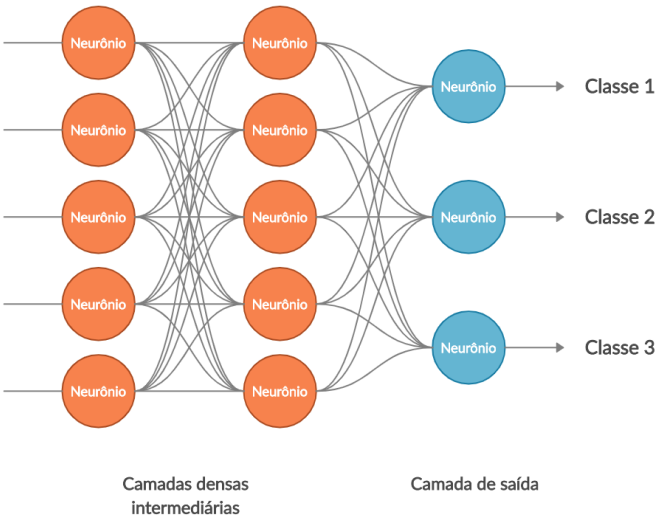


Figura 13. Exemplo de camadas densas com uma saída para 3 classes. Fonte: autoria própria.

[Belhumeur et al. 1997], FEI Faces Database [Thomaz and Giraldi 2010], Face Research Lab London Set [DeBruine and Jones 2017]. Todas as imagens coletadas já estavam no formato JPG, mas com dimensões que variavam entre cada coleção. As dimensões foram ajustadas de acordo com a necessidade de cada algoritmo utilizado. A quantidade e dimensões das imagens juntamente com a quantidade de classes de cada base de dados estão descritas na Tabela 1

Com exceção das bases de dados Faces 1999 (Front), Georgia Tech face database e MIT CBCL Face Recognition Database, todas as coleções possuíam as imagens separadas em subdiretórios para cada classe. Portanto, as bases que não estavam organizadas de tal forma foram reorganizadas para que houvesse o padrão desejado de um diretório para cada coleção com um subdiretório para cada classe da coleção de dados.

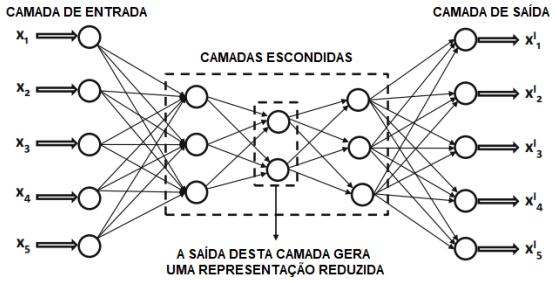
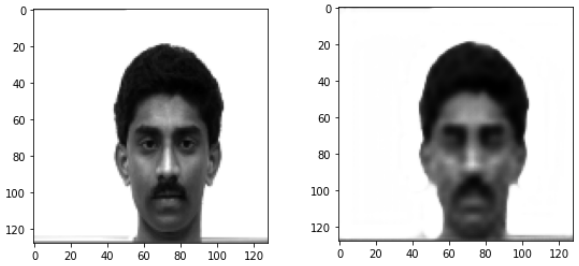


Figura 14. A esquemática básica de um AE. Fonte: Adaptado de [Aggarwal et al. 2018].



(a) Imagem original. (b) Imagem reconstruída através do AEC.

Figura 15. Exemplo de imagem de rosto reconstruído utilizando um AEC. Fonte: autoria própria.

3.2. Pré-Processamento das Imagens

Visando adequar as imagens de rostos para os algoritmos de classificação, foi necessário ajustar as dimensões de altura e largura da imagem de acordo com o algoritmo utilizado. Para as RNCs, todas as imagens foram ajustadas para as dimensões 224 x 224, já para o AEC, 128x128, para que o *bottleneck* seja capaz de representar mais características da imagem original. Para os algoritmos de *matching* e o OC-LBPH, as imagens foram redimensionadas para o tamanho mediano das imagens de cada coleção de dados.

Além disso, as imagens foram convertidas em escala de cinza para os algoritmos de *matching* e AMUC, reduzindo o espaço de cores de 3 (RGB) para apenas 1. O mesmo não foi feito para as RNCs, que, por já criarem filtros necessários para cor, caso seja relevante na discriminação das classes, mantiveram um espaço de cores de 3 posiçoes.

3.3. Algoritmos e Configurações

Nesta subseção são apresentados os parâmetros e configurações utilizados nos experimentos com os diferentes algoritmos da literatura considerados para este trabalho e para o algoritmo proposto.

Tabela 1. Informações das coleções de rostos rotulados utilizadas. Fonte: autoria própria.

Bases de dados			
Nome	Quantidade de imagens	Quantidade de classes	Dimensão das imagens (largura x altura)
MIT CBCL Face Recognition Database	59	10	768 x 576 (15 imagens) 2048 x 1536 (44 imagens)
Faces 1999 (Front)	445	26	896 x 592
Faces 94	3043	152	180 x 200
Faces 95	1440	72	180 x 200
Faces 96	3016	152	196 x 196
Grimace	360	18	180 x 200
Georgia Tech Face Database	750	50	640 x 480
Yale Face Database	165	15	320 x 243
FEI Faces Database	2800	200	640 x 480
Face Research Lab London Set	510	102	1350 x 1350

3.3.1. Algoritmos da Literatura

Os algoritmos de *matching*, descritos na Seção 2.2, e seus parâmetros são:

- **Eigenfaces, Fisherfaces, LBPH:** foram utilizados os valores padrões da biblioteca OpenCV.

Em relação ao AMMC, foram considerados modelos de RNC pertencentes à biblioteca TensorFlow [Abadi et al. 2015]. Foi utilizada a arquitetura de todos os modelos com apenas uma camada densa ao final, com a quantidade de neurônios sendo referente à quantidade de classes da coleção de dados. Os 6 modelos de RNC utilizados nos experimentos foram:

- **Xception** [Chollet 2017].
- **VGG19** [Simonyan and Zisserman 2014].
- **ResNet50V2** [He et al. 2016].
- **InceptionV3** [Szegedy et al. 2016b].
- **InceptionResNetV2** [Szegedy et al. 2016a].
- **MobileNetV2** [Sandler et al. 2018].

Para a camada de entrada das RNCs foram adotadas matrizes de 3 dimensões (altura x largura x espaço de cores), sendo elas 224x224x1 para a coleção de dados Yale Face Database, cujas imagens são originalmente em escala de cinza, e 224x224x3 para as demais coleções. Para a camada de saída foi adotada uma camada densa com função de ativação sigmoid [Narayan 1997], cuja quantidade de neurônios corresponde ao número de classes da coleção de dados a ser classificada. Todos os modelos foram compilados considerando o otimizador Adam [Okewu et al. 2019], com a função de perda *sparse categorical cross entropy* [Zhang and Sabuncu 2018] e a métrica acurácia (Seção 3.4).

3.3.2. Algoritmos Propostos para uso Neste Trabalho

Para este trabalho, foram propostos dois algoritmos de AMUC para o RF, sendo são eles: (i) o AEC; e (ii) *One-Class Local Binary Patterns Histogram* (OC-LBPH). Ambos podendo aprender novas classes para classificação sem a necessidade de retreinamento de todo o modelo.

A arquitetura de AEC foi construída utilizando as seguintes camadas da biblioteca TensorFlow, com as informações resumidas na Tabela 2:

- **InputLayer:** camada de entrada do AE. Para esta camada foram consideradas entradas com dimensões 128x128x1, ou seja, em escala de cinza.
- **Conv2D:** camada convolucional 2D. Foram utilizadas 4 camadas Conv2D. A primeira com 32 neurônios, as duas seguintes com 64 e a última com 1. A função de ativação utilizada foi a *sigmoid* na última camada e ReLU [Aggarwal et al. 2018] nas demais.
- **MaxPooling2D:** camada que realiza a opreação de *max-pooling* em 2D. Foram utilizadas 2 camadas MaxPooling2D com a janela padrão do TensorFlow (2x2).
- **Flatten:** camada que "achata"os valores da camada anterior em uma única dimensão.
- **Dense:** camada densa. Foi utilizada uma camada com 1024 neurônios com ativação softmax [Goodfellow et al. 2016] como bottleneck do AE.
- **Reshape:** camada que remodela sua entrada na forma fornecida. Foi responsável por remodelar a saída da camada densa no formato 32x32x1.
- **Conv2DTranspose:** camada convolucional 2D transposta, também chamada de deconvolucional. Foram utilizadas 3 camadas Conv2DTranspose, as duas primeiras com 64 neurônios e a terceira com 32. Todas com ativação ReLU.
- **BatchNormalization:** camada para normalização e ativação de sua entrada. Foram utilizadas 2 camadas BatchNormalization, uma após cada camada Conv2DTranspose.

O algoritmo OC-LBPH é baseado no algoritmo LBPH (Seção 2.2.3). Seu funcionamento é igual ao descrito na Seção 3.3.1, entretanto, a classificação de um indivíduo para o RF não é dada apenas pela classe do vizinho mais próximo ao exemplo a ser classificado. Nesta abordagem foi considerada a distância do vizinho mais próximo de acordo com limiares definidos automaticamente durante a etapa de treinamento do modelo.

As abordagens de *matching* e AMMC retornam a classe predita, portanto, pode-se utilizar a saída do próprio algoritmo como resposta. Entretanto, a interpretação é diferente nas abordagens de AMUC. Para o AEC e OC-LBPH, foram definidos limiares que denotaram se uma classe pertencia à classe de interesse (distância igual ou inferior ao limiar) ou não (distância superior ao limiar). As distâncias consideradas para cada algoritmo foram:

- **AEC:** Distância euclidiana [Tan et al. 2019] entre as matrizes das imagens de entrada e de saída.
- **OC-LBPH:** Distância euclidiana entre os histogramas de LBP da imagem de entrada e seu vizinho mais próximo.

Tabela 2. Resumo do modelo de AEC. Fonte: autoria própria.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 128, 128, 1)]	0
conv2d (Conv2D)	(None, 126, 126, 32)	320
max_pooling2d (MaxPooling2D)	(None, 63, 63, 32)	0
conv2d_1 (Conv2D)	(None, 61, 61, 64)	18496
max_pooling2d_1 (MaxPooling2	(None, 30, 30, 64)	0
conv2d_2 (Conv2D)	(None, 28, 28, 64)	36928
flatten (Flatten)	(None, 50176)	0
dense (Dense)	(None, 1024)	51381248
reshape (Reshape)	(None, 32, 32, 1)	0
conv2d_transpose (Conv2DTran	(None, 64, 64, 64)	640
batch_normalization (BatchNo	(None, 64, 64, 64)	256
conv2d_transpose_1 (Conv2DTr	(None, 128, 128, 64)	36928
batch_normalization_1 (Batch	(None, 128, 128, 64)	256
conv2d_transpose_2 (Conv2DTr	(None, 128, 128, 32)	18464
conv2d_3 (Conv2D)	(None, 128, 128, 1)	289

Dadas duas matrizes, uma de entrada (E) e uma de saída (S), distância euclidiana, considerada em ambas as abordagens AMUC para o cálculo de dissimilaridade, é dada por:

$$distância = \sqrt{\sum (E - S)^2}$$

O cálculo dos limiares desses algoritmos foi baseado na lista de distâncias calculadas nos conjuntos de treino das classes. Para cada algoritmo, o cálculo foi feito da seguinte forma:

- **AEC:** O modelo, após treinado, gerou uma saída para cada imagem do conjunto de treino. Foi calculada a distância de cada imagem com sua saída do modelo para obter a lista de distâncias.
- **OC-LBPH:** Foi adotado o procedimento *leave-one-out* (LOU) [Sammut and Webb 2010] para obter as distâncias de cada imagem do conjunto de treino com cada uma das outras, resultando na lista de distâncias desejada.

3.4. Validação e Avaliação

Para obter as performances de classificação dos algoritmos e poder compará-los foi utilizado o procedimento de validação cruzada, conhecido como *k-fold cross validation* (KFCV) [Tan et al. 2019]. Nesta abordagem o conjunto de dados é dividido em k pastas, sendo uma pasta utilizada para testar o algoritmo e o restante para treiná-lo. Em todas as abordagens, foi considerado $k = 5$ para para as coleções de dados que possuíam classes com menos de 10 exemplos, e para as demais coleções, foi considerado $k = 10$.

A partir da lista de distâncias de cada algoritmo AMUC, foram definidos os limiares para decidir se a imagem de teste era pertencente à classe de interesse ou não. Sendo a média da lista de distâncias μ e o desvio padrão σ, foi utilizada a estratégia 6σ [Muir 2005] para a definição dos limiares, listados a seguir:

1. Menor distância
2. Maior distância
3. μ
4. $\mu + \sigma$
5. $\mu - \sigma$
6. $\mu + 2 * \sigma$
7. $\mu - 2 * \sigma$
8. $\mu + 3 * \sigma$
9. $\mu - 3 * \sigma$

Para todos os algoritmos utilizados, a medida de performance considerada foi a acurácia. Esta medida corresponde à razão entre a quantidade de classificações corretas e a quantidade total de classificações realizadas pelo algoritmo.

$$Acurácia = \frac{Quantidade\ de\ classificações\ corretas}{Quantidade\ total\ de\ classificações}$$

Para os resultados e comparações finais, foram considerados a média das acurácias e os respectivos desvios padrões, obtidos com as avaliações dos algoritmos nas diferentes pastas de teste geradas através do procedimento KFCV.

4. Resultados

Nesta Seção são apresentados os resultados obtidos após a execução dos experimentos com os algoritmos e parâmetros considerados. Os resultados estão dispostos nas seguintes Subseções: (i) resultados dos algoritmos da literatura; (ii) resultados dos algoritmos propostos; e (iii) comparação geral. Todas as tabelas com os resultados numéricos dos gráficos apresentados nas subseções a seguir, juntamente com os desvios padrões, são apresentados no Apêndice A.

4.1. Resultados dos Algoritmos da Literatura

Na Tabela 3 são apresentados os resultados da média de acurácia e, entre parênteses, o desvio padrão da execução dos algoritmos de *matching* em cada coleção de dados. Na Figura 16 é apresentada uma comparação dos resultados destes algoritmos. No geral, o melhor desempenho foi obtido pelo algoritmo LBPH, possuindo um desempenho superior aos outros dois algoritmos em 7 das 10 coleções de dados.

Quanto às RNCs, na Tabela 4 e na Figura 17 são apresentados os resultados obtidos pelas diferentes arquiteturas em cada coleção de dados. Foi observado que as RNs Xception e InceptionResNetV2 se sobressaíram, obtendo os melhores resultados em 3 coleções cada. Vale ressaltar que, quando comparadas apenas as redes Xception e InceptionResNetV2, a RNC Xception possui os melhores resultados em 7 das 10 coleções.

Por fim, quando comparados os resultados das duas melhores abordagens de cada método da literatura, RNC e *matching* (Figura 18), observa-se os melhores resultados pertencendo à RNC Xception em 8 das 10 coleções de dados.

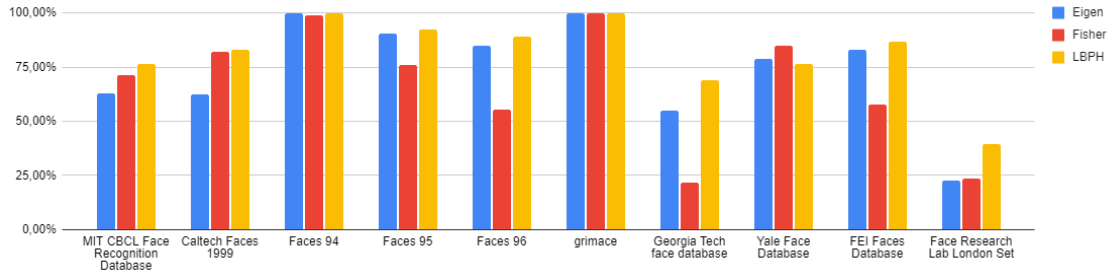


Figura 16. Acurácia dos algoritmos de matching. Fonte: autoria própria.

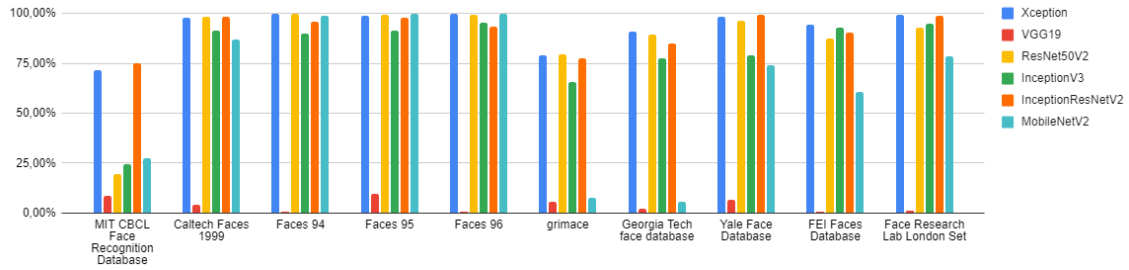


Figura 17. Acurácia das RNCs. Fonte: autoria própria.

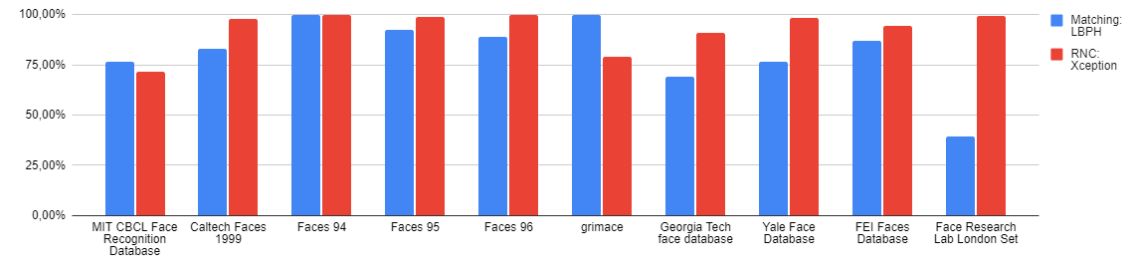


Figura 18. Xception e LBPH. Fonte: autoria própria.

4.2. Resultados dos Algoritmos Propostos

Na Tabela 5 são apresentados os resultados de acordo com cada limiar adotado para a classificação com o AEC. A Figura 19 mostra uma comparação dos resultados dos limiares em um gráfico. A partir dos dados, é possível verificar que o limiar $\mu - 3 * \sigma$ apresentou os melhores resultados em todas as bases de dados.

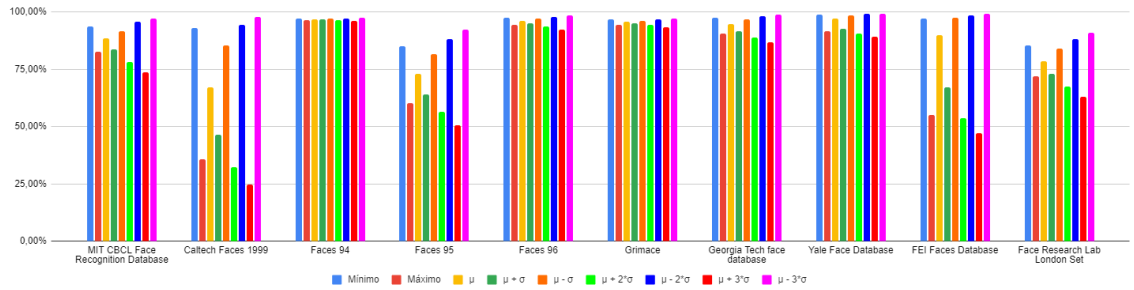


Figura 19. Acurácia dos limiares do AEC. Fonte: autoria própria.

Já em relação ao algoritmo OC-LBPH, os resultados da média de acurácia e desvio padrão dos diferentes limiares testados estão presentes na Tabela 6 e na Figura 20. Pode-se verificar que o limiar $\mu - 3 * \sigma$, assim como no AEC, obteve a melhor performance, possuindo a melhor média de acurácia em 4 das 10 coleções de dados.

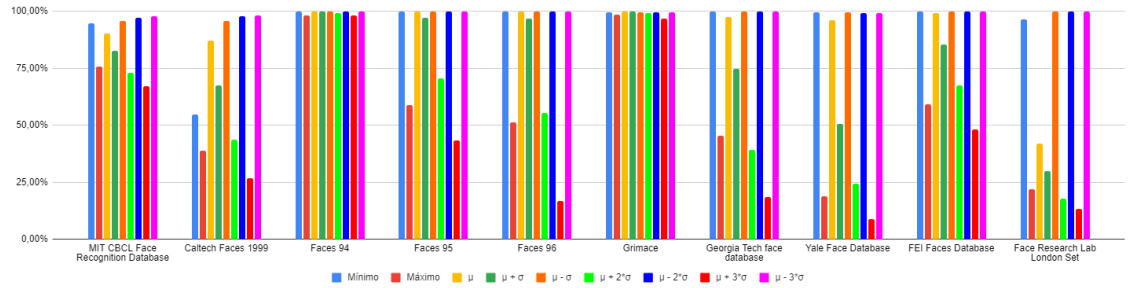


Figura 20. Acurácia dos limiares do OC-LBPH. Fonte: autoria própria.

Quando comparados os melhores limiares dos dois algoritmos de AMUC propostos (Figura 21), pode-se observar uma superioridade do algoritmo OC-LBPH sobre o AEC, com uma acurácia média superior em todas coleções de dados.

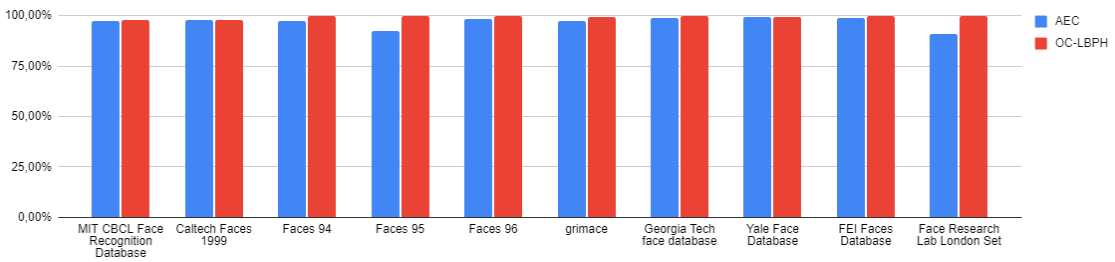


Figura 21. Melhor limiar dos algoritmos AEC e OC-LBPH. Fonte: autoria própria.

4.3. Comparação Geral

Ao comparar os resultados da melhor RNC, algoritmo de *matching*, limiar do AEC e limiar do OC-LBPH, como apresentado na Figura 22, observa-se que o algoritmo proposto OC-LBPH obteve os melhores resultados em 8 das 10 coleções de dados.

O tempo necessário e recursos utilizados no algoritmo OC-LBPH para a inserção de um novo usuário são consideravelmente menores em relação ao AEC e às RNCs, visto que este não utiliza redes neurais para o aprendizado do modelo de classificação. Além disso, outras definições de limiares de classificação podem fazer diferença nos resultados, aprimorando a acurácia do RF. Por outro lado, o uso dos AEC também pode ser aprimorado. Além de definições nos limiares, mudanças na arquitetura do AEC podem aprimorar sua acurácia, fazendo-o destacar-se ainda mais.

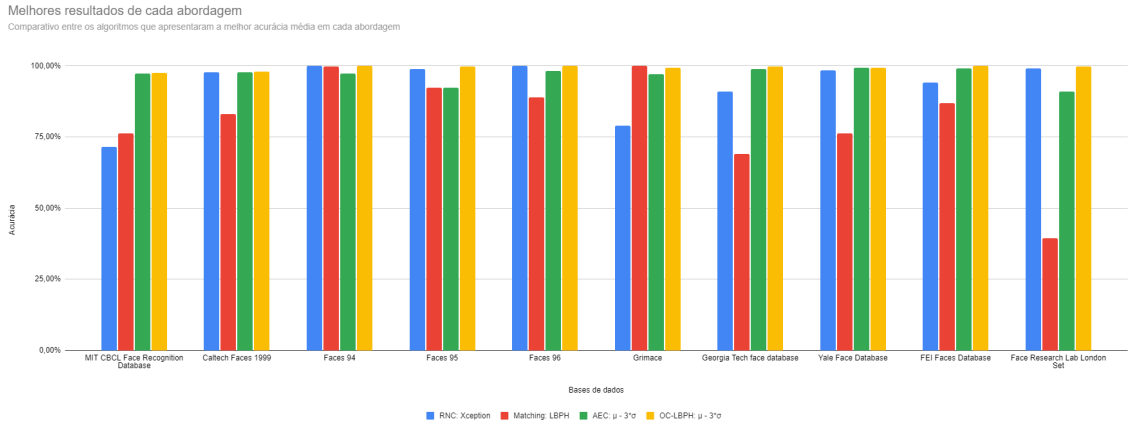


Figura 22. O melhor resultado de cada abordagem. Fonte: autoria própria.

5. Conclusões e Trabalhos Futuros

Este trabalho propôs-se a analisar soluções para lacunas no RF referentes à classificação de indivíduos na ausência de contra-exemplos e ao custo computacional de continuamente retreinar modelos de classificação ao inserir novos usuários. Com os resultados obtidos, observou-se que, no geral, os algoritmos propostos de AMUC obtiveram resultados competitivos e com performance superior à abordagem de AMMC das RNCs e dos algoritmos de matching próprios para RF, podendo assim ser interessante o uso dos mesmos em situações práticas. Aplicações com constante inserções de usuários, utilizando o algoritmo OC-LBPH, teriam facilidade para inserir novos usuários, visto que o treinamento seria praticamente imperceptível e não haveria a necessidade de retreinar todo o modelo para todas as classes. Além disso, também poderia ser aplicado para a autenticação de dispositivos pessoais, como celulares, onde não há a presença de contra-exemplos.

Com trabalhos futuros pretende-se testar outras técnicas para extração de características de imagens e diferentes arquiteturas de AE e RN, como as Redes Neurais Generativas. Nas coleções utilizadas, as variações presentes são de ângulo da face, iluminação, posição do rosto na imagem e expressões faciais. Dado isto, há a intenção de montar uma base de dados em que existam outras variações na face dos indivíduos, como presença e ausência de pelos faciais, uso de bonés, óculos escuros, entre outros.

Além disso, pretende-se desenvolver um *software* de controle de acesso para o Laboratório de Inovação e Engenharia de Software da UFMS/CPTL (LivES).

Referências

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Abate, A. F., Nappi, M., Riccio, D., and Sabatino, G. (2007). 2d and 3d face recognition: A survey. *Pattern recognition letters*, 28(14):1885–1906.

Aggarwal, C. C. (2018). *Machine learning for text*. Springer.

Aggarwal, C. C. et al. (2018). *Neural networks and deep learning*. Springer.

Ahonen, T., Hadid, A., and Pietikäinen, M. (2004). Face recognition with local binary patterns. In *European Conference on Computer Vision*, pages 469–481. Springer.

Alam, S., Sonbhadra, S. K., Agarwal, S., and Nagabhushan, P. (2020). One-class support vector classifiers: A survey. *Knowledge-Based Systems*, page 105754.

Belhumeur, P. N., Hespanha, J. P., and Kriegman, D. J. (1997). Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on pattern analysis and machine intelligence*, 19(7):711–720.

Boureau, Y.-L., Ponce, J., and LeCun, Y. (2010). A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 111–118.

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58.

Cheng, X., Chen, Y., Tao, Y., Wang, C., Kim, M., and Lefcourt, A. (2004). A novel integrated pca and fld method on hyperspectral image feature extraction for cucumber chilling damage inspection. *Transactions of the ASAE*, 47(4):1313.

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1251–1258.

DeBruine, L. and Jones, B. (2017). Face research lab london set.

Duda, R. O., Hart, P. E., and Stork, D. G. (2012). *Pattern classification*. John Wiley & Sons.

Fisher, R. (1936). ªthe use of multiple measures in taxonomic problems. º *Ann. Eugenics*, 7:179–188.

Forsyth, D. A. and Ponce, J. (2002). *Computer vision: a modern approach*. Prentice Hall Professional Technical Reference.

Gonzalez, R. and Woods, R. (2018). *Digital Image Processing*. Pearson.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*, volume 1. MIT press Cambridge.

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J., et al. (2018). Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377.

Guo, G. and Zhang, N. (2019). A survey on deep learning based face recognition. *Computer Vision and Image Understanding*, 189:102805.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645. Springer.

Kemmler, M., Rodner, E., Wacker, E.-S., and Denzler, J. (2013). One-class classification with gaussian processes. *Pattern Recognition*, 46(12):3507–3518.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Larochelle, H., Erhan, D., Courville, A., Bergstra, J., and Bengio, Y. (2007). An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th International Conference on Machine Learning*, pages 473–480.

Li, S. Z. and Jain, A. K., editors (2011). *Handbook of Face Recognition*. Springer London.

Li, X.-L., Yu, P. S., Liu, B., and Ng, S.-K. (2009). Positive unlabeled learning for data stream classification. In *Proceedings of the 2009 SIAM International Conference on Data Mining*, pages 259–270. SIAM.

Max Roser, H. R. and Ortiz-Ospina, E. (2015). Internet. *Our World in Data*. https://ourworldindata.org/internet. (Último acesso em 03/11/2020).

Muir, A. (2005). *Lean Six Sigma Statistics: Calculating Process Efficiencies in Transactional Project: Calculating Process Efficiencies in Transactional Project*. McGraw Hill Professional.

Narayan, S. (1997). The generalized sigmoid activation function: Competitive supervised learning. *Information sciences*, 99(1-2):69–82.

Nefian, A. V. (2013). Georgia tech face database.

Ng, A. et al. (2011). Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19.

Ojala, T., Pietikäinen, M., and Harwood, D. (1996). A comparative study of texture measures with classification based on featured distributions. *Pattern recognition*, 29(1):51–59.

Okewu, E., Adewole, P., and Sennaike, O. (2019). Experimental comparison of stochastic optimizers in deep learning. In *International Conference on Computational Science and Its Applications*, pages 704–715. Springer.

Ruff, L., Vandermeulen, R., Goernitz, N., Deecke, L., Siddiqui, S. A., Binder, A., Müller, E., and Kloft, M. (2018). Deep one-class classification. In *International Conference on Machine Learning*, pages 4393–4402.

Sammut, C. and Webb, G. I., editors (2010). *Leave-One-Out Cross-Validation*, pages 600–601. Springer US, Boston, MA.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520.

Seyfioğlu, M. S., Özbayoğlu, A. M., and Gürbüz, S. Z. (2018). Deep convolutional autoencoder for radar-based classification of similar aided and unaided human activities. *IEEE Transactions on Aerospace and Electronic Systems*, 54(4):1709–1723.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Spacek, L. (2008). Description of the collection of facial images. *University of Essex, United Kingdom. Available at http://cswww. essex. ac. uk/mv/allfaces/index. html (Último acesso em 18/02/2020)*.

Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. (2016a). Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv preprint arXiv:1602.07261*.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016b). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826.

Tan, P., Steinbach, M., Karpatne, A., and Kumar, V. (2019). *Introduction to Data Mining*. What's New in Computer Science Series. Pearson.

Tax, D. M. J. (2002). One-class classification: Concept learning in the absence of counter-examples.

Thomaz, C. E. and Giraldi, G. A. (2010). A new ranking method for principal components analysis and its application to face image analysis. *Image and vision computing*, 28(6):902–913.

Turk, M. and Pentland, A. (1991). Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1):71–86.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine learning*, pages 1096–1103.

Wang, T., Wu, D. J., Coates, A., and Ng, A. Y. (2012). End-to-end text recognition with convolutional neural networks. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 3304–3308. IEEE.

Weaver, A. C. (2006). Biometric authentication. *Computer*, 39(2):96–97.

Weber, M. (1999). Faces 1999 (front). computational vision at caltech.

Wechsler, H., Phillips, J. P., Bruce, V., Soulie, F. F., and Huang, T. S. (2012). *Face recognition: From theory to applications*, volume 163. Springer Science & Business Media.

Weyrauch, B., Heisele, B., Huang, J., and Blanz, V. (2004). Component-based face recognition with 3d morphable models. In *2004 Conference on Computer Vision and Pattern Recognition Workshop*, pages 85–85. IEEE.

Yegnanarayana, B. (2009). *Artificial neural networks*. PHI Learning Pvt. Ltd.

Zhang, Z. and Sabuncu, M. (2018). Generalized cross entropy loss for training deep neural networks with noisy labels. In *Advances in neural information processing systems*, pages 8778–8788.

Zhao, W., Chellappa, R., Phillips, P. J., and Rosenfeld, A. (2003). Face recognition: A literature survey. *ACM computing surveys (CSUR)*, 35(4):399–458.

A. Tabelas de Resultados

Tabela 3. Acurácia e desvio padrão dos algoritmos de matching. Fonte: autoria própria.

Matching - Resultados			
	Eigen	Fisher	LBPH
MIT CBCL Face Recognition Database	62,73% (9,33%)	71,21% (15,05%)	76,36% (13,52%)
Caltech Faces 1999	62,20% (8,12%)	82,02% (3,25%)	83,14% (4,86%)
Faces 94	99,90% (0,16%)	99,01% (1,34%)	99,84% (0,28%)
Faces 95	90,28% (1,96%)	75,83% (2,38%)	92,29% (1,71%)
Faces 96	84,78% (2,30%)	55,17% (2,65%)	88,86% (2,61%)
Grimace	99,72% (0,88%)	99,72% (0,88%)	100,00% (0,00%)
Georgia Tech face database	55,07% (5,66%)	21,73% (6,83%)	68,93% (6,29%)
Yale Face Database	78,79% (4,79%)	84,85% (6,43%)	76,37% (7,23%)
FEI Faces Database	82,88% (2,51%)	57,93% (2,15%)	86,79% (2,33%)
Face Research Lab London Set	22,75% (3,69%)	23,73% (3,21%)	39,41% (5,30%)

RNC - Resultados						
	Xception	VGG19	ResNet50V2	InceptionV3	InceptionResNetV2	MobileNetV2
MIT CBCL Face Recognition Database	71,50% (8,05%)	8,65% (0,53%)	19,48% (4,50%)	24,60% (7,14%)	74,81% (12,39%)	27,45% (9,01%)
Caltech Faces 1999	97,68% (2,84%)	4,20% (0,54%)	98,10% (0,48%)	91,32% (6,96%)	98,52% (3,01%)	86,76% (10,91%)
Faces 94	99,95% (0,17%)	0,65% (0,00%)	99,98% (0,05%)	89,74% (6,27%)	96,01% (5,95%)	98,86% (3,13%)
Faces 95	98,97% (3,15%)	9,80% (6,39%)	99,10% (2,76%)	91,49% (5,61%)	97,60% (3,34%)	99,98% (0,03%)
Faces 96	99,93% (0,11%)	0,66% (0,01%)	99,05% (2,96%)	95,26% (6,31%)	93,47% (7,53%)	99,95% (0,16%)
Grimace	79,08% (6,38%)	5,55% (0,00%)	79,63% (9,53%)	65,83% (6,46%)	77,44% (10,12%)	7,50% (1,28%)
Georgia Tech face database	91,00% (3,79%)	2,04% (0,18%)	89,44% (5,65%)	77,58% (5,97%)	84,80% (2,11%)	5,50% (1,30%)
Yale Face Database	98,40% (3,03%)	6,55% (0,56%)	96,45% (3,86%)	78,99% (8,93%)	99,16% (2,03%)	74,10% (4,77%)
FEI Faces Database	94,15% (3,92%)	0,51% (0,05%)	87,57% (7,24%)	92,61% (2,98%)	90,46% (5,17%)	60,64% (11,26%)
Face Research Lab London Set	99,15% (0,39%)	0,98% (0,00%)	92,98% (7,20%)	94,74% (6,95%)	98,66% (0,57%)	78,35% (4,33%)

Tabela 4. Acurácia e desvio padrão das RNCs. Fonte: autoria própria.

Tabela 5. Acurácia e desvio padrão dos limiares do AEC. Fonte: autoria própria.

Autoencoder Convolucional - Resultados								
	Mínimo	Máximo	μ	$\mu+\sigma$	$\mu-\sigma$	$\mu+2*\sigma$	$\mu-2*\sigma$	$\mu+3*\sigma$	$\mu-3*\sigma$
MIT CBCL Face Recognition Database	93,58% (9,27%)	82,73% (17,70%)	88,41% (11,92%)	83,79% (16,86%)	91,59% (9,24%)	78,10% (18,36%)	95,63% (8,17%)	73,69% (19,55%)	97,24% (6,57%)
Caltech Faces 1999	93,02% (12,14%)	35,72% (20,98%)	67,18% (14,16%)	46,53% (17,37%)	85,39% (11,70%)	32,34% (20,84%)	94,42% (10,57%)	24,63% (22,37%)	97,64% (10,13%)
Faces 94	97,04% (6,85%)	96,37% (10,35%)	96,73% (8,32%)	96,55% (9,35%)	96,90% (7,46%)	96,37% (10,71%)	97,08% (6,79%)	96,17% (11,55%)	97,25% (6,34%)
Faces 95	84,96% (9,87%)	60,24% (17,73%)	72,76% (13,41%)	63,98% (16,44%)	81,43% (10,95%)	56,47% (19,32%)	88,15% (9,42%)	50,35% (21,68%)	92,29% (8,41%)
Faces 96	97,44% (9,25%)	94,14% (17,59%)	96,00% (12,10%)	94,86% (15,11%)	96,98% (10,07%)	93,61% (18,19%)	97,74% (8,97%)	92,33% (20,56%)	98,28% (8,00%)
Grimace	96,56% (7,72%)	94,18% (15,93%)	95,55% (11,63%)	94,88% (15,02%)	96,09% (9,02%)	94,18% (17,73%)	96,70% (6,84%)	93,39% (19,31%)	97,06% (5,04%)
Georgia Tech face database	97,55% (9,19%)	90,48% (16,93%)	94,68% (11,63%)	91,48% (14,33%)	96,77% (10,03%)	88,86% (17,58%)	98,06% (9,10%)	86,56% (20,37%)	98,79% (8,30%)
Yale Face Database	98,93% (13,54%)	91,45% (25,14%)	97,19% (19,55%)	92,58% (23,51%)	98,59% (12,97%)	90,49% (26,12%)	99,08% (8,13%)	89,04% (27,05%)	99,27% (4,46%)
FEI Faces Database	97,02% (6,80%)	55,13% (17,19%)	89,90% (9,91%)	67,12% (16,48%)	97,38% (7,54%)	53,67% (17,20%)	98,49% (5,89%)	47,21% (16,06%)	99,01% (4,46%)
Face Research Lab London Set	85,35% (14,29%)	71,79% (22,58%)	78,50% (17,92%)	72,86% (21,72%)	83,98% (14,47%)	67,58% (24,44%)	88,19% (12,19%)	63,09% (25,67%)	90,97% (10,24%)

Tabela 6. Acurácia e desvio padrão dos limiares do OC-LBPH. Fonte: autoria própria.

OC-LBPH - Resultados								
	Mínimo	Máximo	μ	$\mu+\sigma$	$\mu-\sigma$	$\mu+2*\sigma$	$\mu-2*\sigma$	$\mu+3*\sigma$	$\mu-3*\sigma$
MIT CBCL Face Recognition Database	94,78% (4,18%)	75,84% (22,04%)	90,10% (23,32%)	82,74% (20,08%)	95,65% (0,68%)	72,94% (18,56%)	96,99% (0,40%)	66,95% (12,77%)	97,61% (0,40%)
Caltech Faces 1999	54,82% (3,78%)	38,74% (37,19%)	87,17% (10,61%)	67,45% (38,01%)	95,74% (3,61%)	43,48% (40,28%)	97,60% (2,94%)	26,69% (37,66%)	97,99% (2,36%)
Faces 94	99,94% (0,04%)	98,13% (5,36%)	99,97% (0,05%)	99,71% (0,10%)	99,94% (0,04%)	98,98% (8,34%)	99,93% (0,04%)	98,08% (26,63%)	99,93% (0,04%)
Faces 95	99,87% (0,03%)	58,70% (19,99%)	99,95% (0,03%)	97,15% (0,14%)	99,87% (0,03%)	70,69% (16,23%)	99,86% (0,03%)	43,26% (32,41%)	99,86% (0,03%)
Faces 96	99,94% (0,08%)	51,36% (23,25%)	99,97% (0,08%)	96,62% (1,66%)	99,95% (0,08%)	55,41% (29,23%)	99,93% (0,08%)	16,68% (37,49%)	99,93% (0,08%)
Grimace	99,48% (0,23%)	98,55% (1,13%)	99,76% (0,26%)	99,90% (0,23%)	99,51% (0,23%)	99,09% (1,98%)	99,43% (0,23%)	96,60% (25,54%)	99,42% (0,23%)
Georgia Tech face database	99,78% (0,04%)	45,49% (28,71%)	97,53% (0,25%)	74,56% (19,41%)	99,74% (0,05%)	39,02% (35,65%)	99,80% (0,03%)	18,35% (30,36%)	99,80% (0,02%)
Yale Face Database	99,42% (0,00%)	18,71% (13,69%)	95,95% (1,58%)	50,42% (20,49%)	99,33% (0,06%)	24,41% (17,64%)	99,30% (0,07%)	8,90% (7,75%)	99,29% (0,03%)
FEI Faces Database	99,96% (0,01%)	59,16% (34,93%)	99,15% (1,74%)	85,38% (25,86%)	99,96% (0,01%)	67,27% (31,44%)	99,95% (0,00%)	47,97% (33,73%)	99,95% (0,00%)
Face Research Lab London Set	96,33% (4,69%)	21,92% (5,18%)	41,90% (9,26%)	29,85% (5,45%)	99,66% (0,48%)	17,81% (5,71%)	99,77% (0,08%)	13,11% (5,55%)	99,79% (0,04%)